CEWES MSRC/PET TR/98-47

TANGO Interactive for Remote Consulting and Group Software Development

by

Remek Trzaska
Scott Klasky
Tomasz Major
Marek Podgorny
Geoffrey C. Fox
David E. Bernholdt

08h00598

Work funded wholly or in part by the DoD High Performance Computing Modernization Program CEWES Major Shared Resource Center through

Programming Environment and Training (PET)

Supported by Contract Number: DAHC 94-96-C0002
Nichols Research

TANGO Interactive for Remote Consulting and Group Software Development

4 June, 1998
Remek Trzaska, Scott Klasky, Tomasz Major,
Marek Podgorny, Geoffrey C. Fox, David E. Bernholdt*
Northeast Parallel Architectures Center
Syracuse University
111 College Place
Syracuse, NY 13244

*Author for Correspondence:
bernhold@npac.syr.edu

Abstract

NPAC's TANGO has been proven to be an excellent tool in distance education and corporate training situations. The environment we developed in NPAC for this project extends the areas of TANGO applications to distance consulting. The set of tools we prepared allows geographically dispersed programmers and application designers to work together on complex short term (consulting) or longer-term (software development) efforts as if they were working at the same location.

Table of Contents

1. Introduction

The goal of this project was to develop, based on NPAC's TANGO framework, a prototype distance consulting system, with abilities to let software developers who work in distributed centers work together as if located in the same place. TANGO is a general collaboration system that utilizes the event distribution concept. It has been implemented as an extension to the Netscape Navigator (and Communicator) web browser. TANGO lets users share the state of various kinds of applications, such as chat, shared web browser, videoconferencing module, whiteboard, and many others. The set of TANGO APIs for most of the popular programming languages makes it easy to create new TANGO applications and port existing software to TANGO. The set of tools developed for this project provides an integrated development environment with strong support for all the code development phases: design, coding, debugging, and results analysis. These capabilities have been provided by the development of an object-oriented whiteboard, integration of the XEmacs editor with TANGO, and use of the Scivis package for results analysis.

Proper design of software is often the key to its quality. Thanks to TANGO's object oriented diagramming tool, this stage of software development now can be shared among participants distributed geographically. The features of this tool include: full collaboration via TANGO system, object paradigm deployed to diagram components, a set of typical diagram components (rectangles, ovals, text boxes, lines, etc.).

Logged on to a TANGO server, software engineers can jointly manipulate source code using the modules we wrote to make the XEmacs editor fully collaborative. Our approach to the integration of XEmacs with TANGO allows users to preserve their individual customizations (choice of colors, fonts, key bindings, spell-checking, keyboard abbreviations, etc., defined in the user's ~/.emacs file) while collaborating. The flexibility of XEmacs, and the fact that it is already well integrated with the typical Unix software development environment makes it an excellent tool to provide many of the capabilities requested by software developers.

XEmacs provides interactive, high level, project rather than file oriented, transparent to a user access to the most popular Unix version management systems: CVS and RCS. The system helps to resolve version conflicts, creates version history and ChangeLog files, makes it easy to prepare differential patches between revisions. CVS itself takes care of proper integration of changes made by different programmers.

Testing and debugging code with standard GNU debugger gdb becomes much easier now, when a group of experts can search for software bugs collaboratively in the module we implemented for XEmacs gdb-mode. The support available in the shared XEmacs gdb-mode we designed and implemented includes everything that is necessary for successful debugging. Participants are able to share output from the debugging of running processes, inspect post-crash memory dump core files, display values of variables, set breakpoints, and trace the code line by line or function by function. All this functionality is available for every participant of the same session, no matter if he/she uses the machine on which the debugger itself is running.

Last but not least, we provide a set of tools for visualization of scientific computation process results. Scivis, the toolkit for interactive visualization provides a reach, user-extensible set of numerical filters and the store-and-animate capability for analyzing code numerical stability or for tracing numerical convergence processes. Visualization module provides animated 2D and 3D charts, including isosurfaces.

All the tools discussed in this report are available for download from the Web. TANGO's webpage is located at http://tango.npac.syr.edu/tango. The object oriented diagramming tool is a standard component of TANGO, and does not require special installation. The modules for shared editing and shared debugging in XEmacs via TANGO can be obtained from http://www.npac.syr.edu/users/remek/cewes/XEmacs-shared.tgz. Scivis has its own webpage at http://kopernik.npac.syr.edu:8888/scivis.

After a short description of the TANGO system we focus on individual components of the remote consulting system we have developed: shared XEmacs editor, shared object oriented diagramming tool, shared debugger, scientific visualization package Scivis. Future possible enhancements to the system are discussed afterwards. User guides to the components of the TANGO distance consulting environment are provided as appendices.

2. TANGO

TANGO Interactive is a web collaboratory. The system extends capabilities of web browsers towards a fully interactive, multimedia, collaborative environment. TANGO is also a framework for building collaboratory systems of arbitrary complexity.

TANGO Interactive is written in Java. Many application modules are implemented as applets. The applets can be loaded when needed and released at any time, ensuring that the system is lean and agile. The applets interact with each other, can control each other's behavior and exchange data, both locally and remotely. TANGO applets can be loaded from anywhere—there is no requirement that all applets or even different instances of the same applet come from the same HTTP server. The TANGO collaboratory server does not need to be co-located with any of the HTTP servers from which application modules are being downloaded. TANGO Interactive is the only collaboratory system we are aware of implementing this flexible and powerful architecture.

TANGO Interactive is very tightly integrated with the Web. One starts the system from a Web browser, which, thereafter, one can continue using. The system connects the user to a TANGO server of user’s choice.

Once in the system, the user can open the various collaboratory applications included to work on projects with partners, take a class at a virtual university, create and use a chat room, videoconference, watch a movie with friends, collectively surf the Web and discuss the contents, or play a network game. Or do all of the above at the same time, in any combination. Few collaboratory systems, public domain or commercial, provide so many applications under a consistent and very simple session and floor control.

TANGO Interactive is an event-sharing system. The system requires that there is a local copy of each application on every workstation. The shared functionality is defined by application, not by the collaboratory runtime. This is in sharp contrast to the most popular commercial collaboratory systems, such as NetMeeting from Microsoft. These systems

implement the shared display model. Shared event systems have a number of advantages over shared display systems, including capability to define shared functionality, to implement independent data views, and to support asynchronous collaboration. TANGO Interactive extensively uses these capabilities without sacrificing advantages of the shared display model: NetMeeting's shared application functionality is fully integrated into the TANGO session control model.

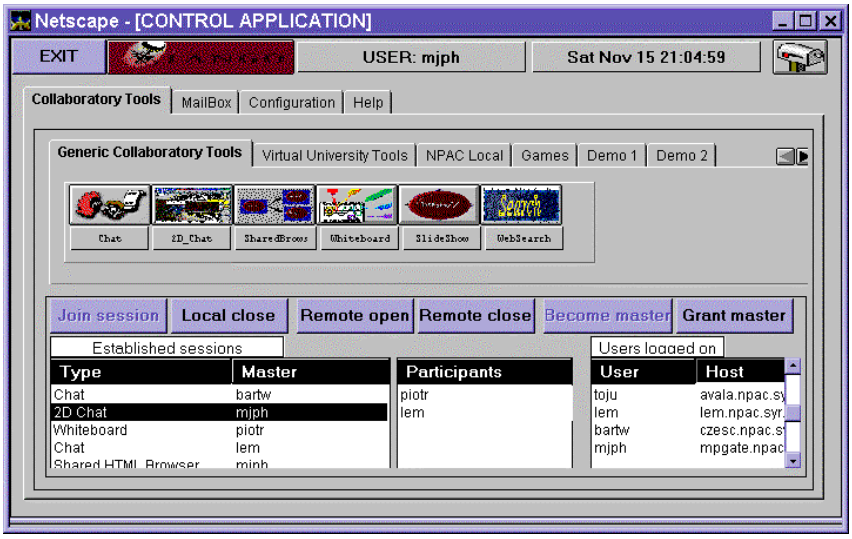

Figure 1. The main TANGO control panel.

The design of the TANGO Interactive protocols is based on the assumption that only the application should define collaboratory functionality. We recognized that collaboratory applications might require functionality that we cannot envision today. Hence, TANGO does not define application protocols. Instead, it provides a reliable, high-performance message passing mechanism, integrated with the session management, which can be easily used to implement any application-level protocols.

TANGO supports all functions of a synchronous collaboratory, including session management, data/event distribution, flexible floor control, and multiple, configurable security levels. The system does not impose any restrictions on the number of concurrent sessions, users, or collaborative applications. Application sessions are created via a mouse click and application instances can be created on local and remote workstations on the fly.

The floor control model allows for both master-master and master-slave relationships between collaborating parties. The actual implementation of the master-slave relationship is left to the application designer, allowing for a variety of floor control models. TANGO collaboratory runtime provides functionality for dynamic modification of the master-slave relationship at any time.

While almost the entire TANGO runtime and most of the applications are written in Java, there is no restriction on the language used to implement TANGO applications modules. At present, Java is not very well suited for applications in domains as high-end 3D visualization, high quality video streaming, and videoconferencing (especially

multimedia encoding). TANGO integrates applications written in C, C++, and LISP just as easily as Java applets.

3. Shared Editing

We decided to prepare a collaborative version of XEmacs, which is an X Window oriented version of GNU Emacs with many enhancements of the original. (X)Emacs is a very popular public domain editor, with extensive support for all phases of the code development cycle. This choice provides powerful customization possibilities, because most of XEmacs submodules are written in Emacs LISP, and are loaded and interpreted dynamically. That is why XEmacs behavior can be modified easily, by replacing its functions and/or extensions of their functionality by means of “advises”—special function wrappers evaluated before or after the actual function (LISP *form*).

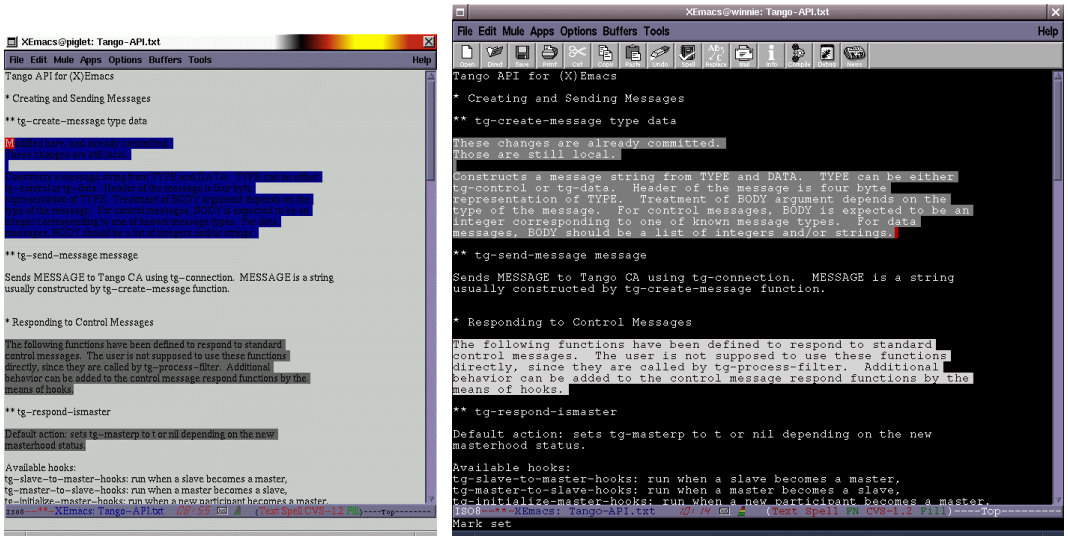

Figure 2. Contents of two shared TANGO Xemacsen.

Collaborative writing systems have been divided into three categories: synchronous, where editing of the same document by different authors takes place at the same time; asynchronous, where only one person controls the entire document at a particular moment; and semi-synchronous, where authors are able to work together on different sections of the same document at the same time. When designing a shared editor with strong support for code development for TANGO, we decided to implement it as a semi-synchronous one. This approach provides reasonable cooperation capabilities while enforcing on collaborators discipline, and is not very demanding as far as network bandwidth goes.

A typical TANGO session of shared XEmacs starts with opening a new or existing file in XEmacs spawned by TANGO demon. The initial content of the file is shared among session participants. When any of the participants wants to edit a section of the file, he/she selects it and requests a lock for this region of text. It is up to the user when to broadcast to other participants the changes he/she has performed. Finally, when the user finishes editing the region, he/she releases it, and others can review it.

4. Shared Object Oriented Diagramming Tool

The TANGO shared object oriented diagramming tool, Wbd, provides functionality similar to the one of Microsoft PowerPoint, although its focus is shifted toward drawing rather than preparing presentations. It offers all of the standard drawing tools available in commercial applications: rectangles, ovals, polygons, arrows, free-forms, text, etc.

Wbd content sharing via TANGO is based on master-master relationship, which means that every participant has equal rights to perform editing. Changes of the state of every object are reflected in other participants' whiteboards immediately, with a few exceptions, such as resizing, moving or changing some object's properties. In these cases, updates are broadcast as soon as the modifying action terminates.

5. Shared Debugging

The TANGO shared debugger extends functionality of traditional tools for tracing programs and fixing bugs to fully collaborative mode. It lets one user start a process on his/her machine under GNU gdb debugger in XEmacs gdb mode, and afterwards grant full control over this process to any participant of the collaboration session. Other participants may run TANGO on any hardware/software platform that supports both XEmacs and TANGO. We refer to the person who initiates the debugging process by issuing the XEmacs command `gdb` as a debug process owner.

For example, a session may be established between a debug process owner running TANGO on an SGI mainframe, and a consulting expert connected from a SUN workstation. Every action of the remote user is monitored in the debug process owner XEmacs buffer, and in emergency debugging process can be killed by him/her instantly. The TANGO XEmacs shared debugger fully supports master/slave changes. Any of the participants at any time can ask to become a master, and if approved, gain control over the debugging process.

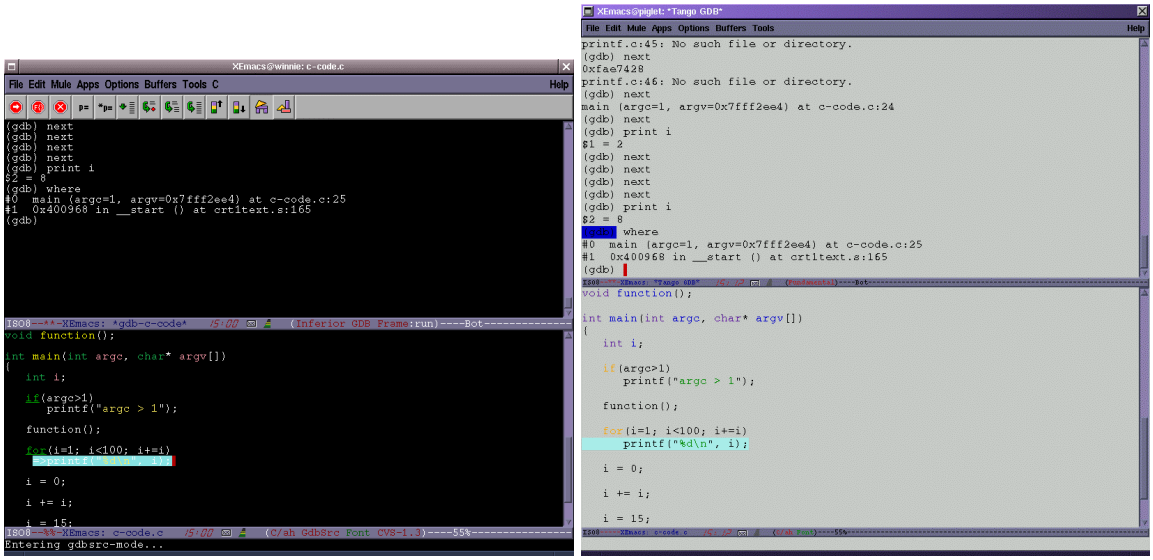

Figure 3. Shared debugging session between two participants. The one on the left is the process owner using SGI O2, the one on the right is a "slave" participant using a PC running Linux.

As far as implementation goes, the TANGO XEmacs shared debugger is integrated with XEmacs gdb-mode. The mechanism of ELISP “advices” allowed us to modify behavior of gdb mode commands without changing their names, so that everyone who is familiar with debugging via (X)Emacs gdb mode is immediately able to use TANGO XEmacs shared debugger. The output of both GNU gdb debugger and the application being debugged is shared among all participants of the TANGO XEmacs shared debugger session. The current master of the session enters commands for the debugger in the TANGO debug buffer.

Whenever the regular XEmacs gdb mode is able to display a source code line corresponding to the current frame it pops up a buffer with this line highlighted. TANGO XEmacs shared debugger provides the same functionality. Source files are transmitted from debug process owner’s filesystem to other session participants and stored there in a temporary directory

6. Scientific Visualization

Scivis is a collaborative scientific data visualization package that will aid researchers from distant, diverse locations to work together in developing scientific codes, providing them with a system to analyze their scientific data. It has been developed in Java. Scivis was designed with focus on two important areas: a collaborative framework from which the scientific data is interpreted and utilized, and a framework, which is customizable to the suit of a particular task and/or scientific group. The Scivis package is not presently integrated with TANGO, but as we investigated the alternatives, it was clearly the most effective way to provide the required capabilities. Integration with TANGO was beyond the scope of this project, but is now underway separately.

Scivis obtains data either from the client via a socket or directly from a file. Data sets may be composed of multiple time slices. The basic visualization features of Scivis include:

- visualize 2-dimensional (x, f(x)), 3-dimensional (x, y, f(x, y)), vector, isosurface data and contour plots,
- step through time slices, and animate them over time, with customizable speed,
- create output in Postscript and GIF formats,
- use customizable color maps, expressed in both RGB and HSV color spaces,

User definable filters are a very important feature of Scivis. Basic filters, provided with the software, include merging of data sets, extraction of a 2-dimensional data from a 3-dimensional data set over the x and y values, extraction of some portion of the data, and sample numerical analysis filters such as cubic spline or least square methods. The user can extend this set by writing Java classes according to the guidelines provided in the Appendix 3.

7. Future Enhancements

Future development of the system might result in several enhancements, which will greatly improve its overall quality and usability. Among them there are the following issues:

- We currently work on porting Scivis to TANGO. This change will let users start the whole environment with a couple of mouse clicks directly from a browser's window, and control collaboration aspects of every application from the same TANGO panel.
- There are some improvements possible as far as asynchronous file transfer for XEmacs shared editor and shared debugger are concerned.
- Currently, shared editor and shared debugger modules for XEmacs cannot coexist in one instance of XEmacs. This forces the user to start two memory and resource consuming processes. We expect that removal of this drawback does not require significant redesign and coding.
- XEmacs built-in interfaces to version control systems could be integrated with the TANGO shared editor module. `tg-edit-commit-region` command might invoke corresponding CVS or RCS command, resulting in storing the patch in the software repository.
- Use of the gdb debugger may be somewhat limiting, but it is unrealistic to attempt to provide TANGO integration for the plethora of proprietary debuggers currently used on HPC systems. One alternative approach, which we are investigating involves integration of a simple "xterm"-like tool into TANGO. This would support text-based debugging and other tools.

8. Conclusions

XEmacs provides vast variety of helpful programmer's tools. The TANGO API for (X)Emacs LISP gives these powerful tools to software engineers who need to communicate with geographically distributed developers in a synchronous manner.

TANGO opens wide areas of possible collaboratory applications: virtual university, corporate training, telemedicine, and with this new set of tools, remote consulting focused on software engineering. We expect that cooperative software design, collaborative source code writing, shared debugging and visualization of the results provided by one centralized environment will change the way software is produced.

9. Acknowledgements

This project was sponsored by the DoD High Performance Computing Modernization Program CEWES Major Shared Resource Center through Programming Environment and Training (PET). Supported by Contract Number DAHC 94-96-C0002, Nichols Research.

The major technology TANGO was initially developed with funding from Rome Laboratory.

Appendix 1. XEmacs Shared Editor and Shared Debugger Modules

Both the TANGO shared editor and TANGO shared debugger modules require XEmacs 19.15 or newer. They should work with FSF Emacs 19.34 or newer, however slight modifications of the code might be required. As of May 1998, XEmacs support for MS Windows 95/NT is still in beta phase, therefore shared editing via TANGO in XEmacs is possible only between UNIX architecture clients. Similarly, TANGO shared debugger is based on (X)Emacs gdb mode, and thus will run only on platforms GNU gdb is supported on, which are all popular UNIX architectures.

However, when a version of XEmacs for MS Windows 95/NT appears, shared editing between collaborators running TANGO on any platform will be possible. Shared debugging will let users inspect applications run on Unix workstations remotely by TANGO users working on PC's. Remote control and debugging of Windows applications will depend on availability of GNU gdb on MS Windows.

It is assumed that users of TANGO shared XEmacs know how to use XEmacs, and are familiar with XEmacs related terms such as region, buffer, face, and alike. For customization of TANGO shared XEmacs, basic knowledge of Emacs LISP is required.

Installation

The whole installation necessary for TANGO XEmacs shared editor and shared debugger modules is just downloading the archive from:

http://www.npac.syr.edu/users/remek/cewes/XEmacs-shared.tgz

and unpacking it to the appropriate location, determined by the value of the `TANGO_APP_ROOT` environment variable. This can be achieved by issuing the following commands:

```
cd $TANGO_APP_ROOT
gunzip –c /tmp/XEmacs-shared.tgz |tar xvf –
```

assuming that the archive has been downloaded to `/tmp/XEmacs-shared.tgz`. No changes to the PATH environment variable are necessary, provided that 'xemacs' command is available. No changes to XEmacs load-path are necessary either—TANGO XEmacs shared modules are capable to determine locations of all ELISP files from the value of `TANGO_APP_ROOT` environment variable.

Collaborative Writing

A typical collaboration session using TANGO shared XEmacs starts with opening a new or existing file and selecting by different users regions each of them is going to edit. After a user gains lock of a region, he/she may edit it as in any text editor. The user decides when to send updates of the region he/she holds lock for. When the user finishes editing the region, he/she releases it, so that others can review it.

Registering Document to Be Shared

Before a document can be edited in a collaborative way, it has to be registered in TANGO XEmacs. The command `tg-edit-register-file` (by default bound to `C-c t r`) is meant to perform this task. It is interactive and takes one argument, which is a name (either relative to current directory or absolute path) of the file to be registered for collaborative editing via TANGO.

Note: If either a non-existing or an empty file is registered, TANGO XEmacs will automatically insert a single newline character at the beginning of the file. This is required to let the user select a region for locking.

Locking Region for Modifications

Initially, just after opening a file, all of its contents is not available for modification. A special command, `tg-edit-add-region` (`C-c t a`) has to be issued on a selected region that will lock this region, register it in TANGO XEmacs, and permit the owner of the lock to perform editing operations. First, `tg-edit-add-region` checks if the selected region is not overlapping any other TANGO regions. If the region selection is legal, read-only property is removed from the region. Then, visual features, such as face and balloon help property are set, and finally an appropriate message is sent to other participants.

Other XEmacs clients in response to this type of message, display the region with special face and set the balloon help property of the region to a text “locked by ownername.”

Modifying Regions

A locked TANGO region is open for modifications by its owner (a user who holds the lock). Those operations include all typical editing functions, such as insertion, deletion, replacements, etc.

A command `tg-edit-delete-region`, bound to `C-c t d`, removes the contents of a TANGO region.

Note: It is crucial for consistency of distributed copies of a shared file not to remove the entire TANGO region with ordinary deletion operations. These operations remove the XEmacs extent, which identifies TANGO region, and thus commit and release operations cannot be performed successfully. TANGO region, once entirely removed from the file cannot be restored by undo command. To remove a TANGO region, use `tg-edit-delete-region` command (`C-c t d`) only.

Committing Changes

To preserve bandwidth and improve performance, TANGO region changes introduced by editing operations are not transmitted to other session participants immediately. When the owner of a TANGO region lock decides to update others’ copies of the shared file, he/she issues a command `tg-edit-commit-region` (`C-c t c`). This command requires the point to be inside a TANGO region belonging to the person who issues it.

Releasing Regions

After a user has finished modification of a region of the shared file, he/she is supposed to release it, so that other participants may gain the lock over any fragment of it. This is accomplished by a command `tg-edit-release-region` (`C-c t l`), which similarly to `tg-edit-commit-region` requires the point to be inside of the region.

Note: a TANGO region has to be committed before it can be released.

Unregistering Files

Since every participant in a shared editing session stores his/her own copy of the shared file, his/her closing of a TANGO buffer does not affect other collaboration parties. The only operation performed by TANGO XEmacs is automatic committing and releasing of all the regions locked by the participant who is closing the TANGO buffer. These actions are performed automatically when any standard XEmacs buffer removal operation is invoked on a TANGO buffer.

Killing XEmacs

TANGO XEmacs substitutes the key that is bound to the command `save-buffers-kill-emacs` with its own function, `tg-exit-emacs`. This command unregisteres all the TANGO buffer the user shares with others participants (what results in losing all region locks). If the user who is about to close XEmacs is the master of the TANGO XEmacs session, a special notifying message is sent to others. After a delay to allow for delivery of this message, the socket connection to the TANGO demon is closed, and the usual process of closing XEmacs is carried on by calling the command `save-buffers-kill-emacs`.

Customization

Most likely, users will want to change the properties of faces used to display a locally locked region and a face used for remotely locked region to adjust them to the background color he/she uses. The defaults for these faces are:

```
(set-face-background 'tg-locked-here-face "NavyBlue")
(set-face-background 'tg-locked-others-face "grey40")
```

Default keybindings for TANGO XEmacs shared editing operations start with C-c t prefix and are following:

```
(global-set-key [(control c) t r] 'tg-edit-register-file)
(global-set-key [(control c) t a] 'tg-edit-add-region)
(global-set-key [(control c) t c] 'tg-edit-commit-region)
(global-set-key [(control c) t d] 'tg-edit-delete-region)
(global-set-key [(control c) t l] 'tg-edit-release-region)
```

These defaults can be changed according to user's preference. To make the changes permanent, appropriate LISP forms should be placed in user's own `.emacs` file.

Shared Debugging

The TANGO XEmacs shared debugger fully supports master/slave changes. Any of the participants at any time can ask to become a master, and if approved, gain control over the debugging process.

Starting a Session

The TANGO XEmacs shared debugger is integrated with XEmacs gdb mode. The mechanism of ELISP "advices" allowed us to modify behavior of gdb mode commands without changing their names, so that everyone who knows how to debug code with help of (X)Emacs gdb mode is immediately able to use TANGO XEmacs shared debugger.

A new debug process is initiated with `M-x gdb` command. This creates a dedicated buffer with output of both GNU gdb program and the application being debugged. The content of this buffer is shared among all participants in the TANGO XEmacs shared debugger session. The current master of the session enters commands for the debugger in this buffer.

Whenever the regular XEmacs gdb mode is able to display a source code line corresponding to the current frame, it pops up a buffer with this line highlighted. The TANGO XEmacs shared debugger provides the same functionality. Source files are transmitted from debug process owner's filesystem to other session participants. When the file is received, TANGO XEmacs shared debugger stores it in a temporary directory defined as `/tmp/TANGO-shared-debugger`.

Pitfalls

Sending commands to debugger process running on a remote machine is initiated by pressing return or enter key. This command always takes the contents of the last line of the buffer, possibly removes the debugger prompt "(gdb)", and sends it to debug process owner. It happens sometimes that the debugger puts in this line some text by itself, such as questions (e.g. "The program is already being run. Restart it? (y on n)"). If the user inserts the answer in the same line, debugger will receive the whole line instead of just the text typed in by the user. In such situations, the user must first start a new line, by pressing `C-u return`, and then answer with single letter `y` or `n`.

TANGO API for XEmacs Lisp

Creating and Sending Messages

- `tg-create-message type data`

Constructs a message string from TYPE and DATA. TYPE can be either `tg-control` or `tg-data`. Header of the message is four-byte representation of TYPE. Treatment of BODY argument depends on the type of the message. For control messages, BODY is

expected to be an integer corresponding to one of known message types. For data messages, BODY should be a list of integers and/or strings.

- `tg-send-message message`

Sends MESSAGE to Tango CA using `tg-connection`. MESSAGE is a string usually constructed by tg-create-message function.

Responding to Control Messages

The following functions have been defined to respond to standard control messages. The user is not supposed to use these functions directly, since they are called by `tg-process-filter`. Additional behavior can be added to the control message respond functions by means of hooks.

- `tg-respond-ismaster`

Default action: sets `tg-masterp` to `t` or `nil` depending on the new mastership status.

Available hooks:

`tg-slave-to-master-hooks`: run when a slave becomes a master,

`tg-master-to-slave-hooks`: run when a master becomes a slave,

`tg-initialize-master-hooks`: run when a new participant becomes a master,

`tg-initialize-slave-hooks`: run when a new participant becomes a slave.

Hooks arguments: none

- `tg-respond-username`

Default action: sets `tg-username-string` to a string received from Tango CA.

Available hooks: `tg-respond-username-hooks`.

Hooks arguments: none (user's name can be determined from a value of `tg-username-string`).

- `tg-respond-mastername`

Default action: sets `tg-mastername-string` to a string received from Tango CA; displays a message stating the master's name in the current XEmacs session.

Available hooks: `tg-respond-mastername-hooks`

Hooks arguments: none (master's name can be determined from a value of `tg-mastername-string).`

- `tg-respond-hostname`

Default action: sets `tg-hostname-string` to a string received from Tango CA; displays a message stating the name of the host the current Tango session is running.

Available hooks: `tg-respond-hostname-hooks`.

Hooks arguments: none (host's name can be determined from a value of `tg-hostname-string).`

- `tg-respond-participants`

Default action: sets `tg-participants-list` to a list of all the participants of the current session of Shared XEmacs; displays a message listing them.

Available hooks: `tg-respond-participants-hooks`.

Hooks arguments: none (list of participants can be taken from `tg-participants-list`).

- `tg-respond-activeusers`

Default action: sets `tg-activeusers-list` to a list of all the users of the current Tango session; displays a message listing them.

Available hooks: `tg-respond-activeusers-hooks`.

Hooks arguments: none (list of active users can be taken from `tg-activeusers-list`).

- `tg-respond-localapps`

Default action: sets `tg-localapps-list` to a list of all the local Tango applications that are running on the machine the Tango session has been started; displays a message listing them

Available hooks: `tg-respond-localapps-hooks`.

Hooks arguments: none (list of local applications can be taken from `tg-localapps-list`).

- `tg-respond-focus`

Default action: raises the current XEmacs frame.

Available hooks: `tg-respond-focus-hooks`.

Hooks arguments: none.

- `tg-respond-participants-and-addresses`

Default action: sets `tg-participants-and-addresses-list` to a list of all the participants of the current Shared XEmacs sessions with the addresses of the hosts they are using to connect to Tango; displays a message listing the content of this list.

Available hooks: `tg-respond-participants-and-addresses-hooks`.

Hooks arguments: none (list of participants and their addresses can be taken from `tg-participants-and-addresses-list`).

Responding to Data Messages

Tango should be notified of all possible data messages and their respective handlers via the value of `tg-data-message-handlers`. This variable should be an alist. Each element should be a dot pair of the form `(value . handler)`, where VALUE determines data message type and is a short integer, and HANDLER is a function to be called when a message of this particular type is delivered from Tango CA. Adding a message type and its handler can be performed by a form similar to:

```
(push '(100 . tg-ee-respond-register-file) tg-data-message-handlers)
```

Each of handlers is provided with the content of the data message as a list (see tg-create-message).

Registering Application

- `tg-register-application id port`

Establishes the connection with Tango CA on the machine determined by `tg-host` (usually the same machine XEmacs is running on), on the port PORT, with application id ID. Resets the value of `tg-string`. The parameters ID and PORT are obtained usually from Tango CA.

Constants

The following constants have been defined

```
(defconst tg-ismaster 1)
(defconst tg-username 2)
(defconst tg-mastername 3)
(defconst tg-hostname 4)
(defconst tg-participants 5)
(defconst tg-activeusers 6)
(defconst tg-localapps 7)
(defconst tg-focus 122)
(defconst tg-participants+addresses 123)
(defconst tg-participants+addresses-rq 124)
(defconst tg-yes 10)
(defconst tg-no 11)
(defconst tg-terminate 12)
(defconst tg-control 30)
(defconst tg-data 31)
```

Appendix 2. Wbd—Object Oriented Drawing Tool

This user guide is also available on the Web, at http://trurl/tomm/deneb/wbd/wbd/help.html.

Wbd's basic functionality is similar to the one of MS PowerPoint, although the focus of the currently available version is on drawing rather than creating presentations. New, custom-made objects can be imported and become fully functional components inside wbd. Objects may be just shapes but also active components (e.g. video player) performing complicated operations while wbd is in the play mode.

Wbd works as a standalone Java application, or as an applet embedded in the Web page, or collaboratively within the Tango collaboratory framework.

Introduction

- buttons in the left hand menu will be called icons
- hints appear 1 second after mouse is stopped
- names of widgets in the text below corresponds to hints or labels on widgets
- the status bar at the bottom of the whiteboard shows status of any action and objects drawn as well as hints
- any coordinates or dimensions are presented following the convention "first vertical (top to bottom) then horizontal (left to right)"
- send remarks, report bugs to Tomasz Major, toma@npac.syr.edu.

Selecting existing object

- to select an object: click it (selection points shall appear)
- to transfer selection to another object:
  - press Tab (focus moves to the next object toward the top layer); or
  - press ShiftTab (focus moves to the previous object)
- to select multiple objects: drag mouse over objects (selected area will be outlined during dragging)

Moving and resizing selected object

- to move: drag the object (do not drag selection points)
- to resize: drag one of the object's selection points

Object creation and deletion

- to delete selected objects: press Delete key or Delete icon
- to delete all and start new page: press New icon

- to create new object: click one of the object icons, click it and:
  - Rectangle: drag the desired shape
  - Oval: drag the bounding box for the oval
  - Polygon: click position of the corners, double click the last one
  - Line/Arrow: drag from the beginning to the head of the arrow
  - FreeForm: drag the desired shape
  - Text: click the position of text
  - TextBlock: drag the width of the bounding box
  - TextBlockWithBullets: click the position or drag the width of the bounding box
  - Foil: click the position or drag the width of the bounding box
  - Foilset: click the position
- some objects might be created automatically:
  - Container: as a result of group/ungroup operation, and when new page is loaded
  - Image: when GIF or JPEG file are imported
- to clone selected objects: press Clone icon
- to import an object: select the location and press Add button

Structural operations in containers

- to group selected objects: press Group icon
- to ungroup selected objects: press Ungroup icon
- to bring a selected object to front of another object: press BringForward icon (each click on the button moves object only one layer)
- to send backward a selected object: press SendBackward icon

Editing selected object

- to edit properties of an object: press CtrlSpace or double click the object (property editor dialog will appear)
- to edit object: press Space (dashed frame will appear around the object), editing is object dependent:
  - Rectangle: none allowed
  - Oval: none allowed
  - Polygon: dragging contour points, creation (press key N) and deletion (key D) of contour points
  - Line/Arrow: dragging beginning and end

- FreeForm: same as polygon
- Text: entering text, popular Emacs key combinations (Alt->, Alt<-, CtrlSpace, AltW, CtrlW, CtrlY, CtrlK), CtrlShiftL/ CtrlShiftC/ CtrlShiftR for alignement and anchoring (left/center/right respectively)
- TextBlock: editing text as in Text object, highlighting, AltN/ AltS/ AltC/ AltK/ AltL for changing font style (normal/strong/cite/keyboard/strong keybord respectively)
- Foil: same as Container
- Foilset: same as Foil
- Image: none allowed
- Container: doing anything as if it were embedded whiteboard (the main panel of the whiteboard is a container) - see Editing Container

- to quit editing: press ShiftSpace (dashed frame will disappear, object will remain selected)

Editing containers

A container is a component that receives special attention because all objects are created in containers and belongs to containers (except the most external one). The key functionality of the whiteboard described in different chapters is in fact the functionality of containers.

Editing of a container is the same as the editing of the main board (the main panel of the whiteboard is also a container). The following key shortcuts are available:

- N/I/C - create new object/import an object/clone selected object
- D/ShiftD - delete selected object/delete all objects
- E - edit properties of selected object or of this container if nothing selected
- L/ShiftL - bring forward/send backward selected object
- G/ShiftG - group selected objects/ungroup selected object
- M/ShiftM - copy selected object to clipboard/paste from clipboard to the center of this container

Importing and exporting objects

- GIF and JPEG images, files in wbd format and plain ASCII text can be imported
- any whiteboard creation can be saved in wbd format (links to images) or as a postscript file
- to import: select location and press Add or Replace button, or press Open icon
- to export: press Save or Print icon

Sending updates in Tango

Note: This whiteboard can be used independently of the TANGO collaboratory system.

- the system works in master-master mode:
- each participant have equal rights to perform editing
- initialization data is delivered by one participant
- updates are not always sent immediately:
    - creating - after creation is completed
    - moving, resizing - when mouse raised
    - modifying properties - when Apply or OK buttons are pressed in a Property Editor
    - editing - after exiting editing mode
    - other actions - immediately
- some control over Tango session is provided by the Tango dialog

Most important key combinations and shortcuts

- Tab transfers focus to the next object (toward the top layer)
- ShiftTab transfers focus to the previous object
- Space enter editing mode (one object must be selected)
- ShiftSpace quit editing mode
- CtrlSpace show property editor (one object must be selected)
- Delete delete selected object

Known bugs

- printing of the whiteboard contents is problematic
- do not change the position or size of the main container

Appendix 3. Scivis—Scientific Visualization Package

The user guide for Scivis has been published as a separate document, available on the Web at http://kopernik.npac.syr.edu:8888/scivis/guide.html.